VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY

UNIVERSITY OF SCIENCE

FACULTY OF INFORMATION TECHNOLOGY

REPORT

Lab 2: Adversarial Search

Subject: Fundamentals of Artificial Intelligence

Trịnh Thế Sơn - 20127617

UNDER THE GUIDANCE OF

Lecture: Pham Trong Nghia
TA: Nguyen Thai Vu

I) PROJECT INTRODUCTION

Problem description

Students implement a tic-tac-toe game (Vietnamese: trò chơi caro). For simplicity, students just need to implement 3x3 maps. Student will control player 1, Computer will control player 2 (and vice versa). - Students choose any adversarial search. Using that adversarial search to find the optimal path, which will help the Computer to win this game.
Students must implement the interface of tic-tac-toe game. Note:

- You can use pygame or tkinter (python library), those libraries are so easy to learn and use.
- The main purpose of this project is learning Adversarial search, please do not focus on application or interface (just easy to look are enough).

II) MINIMAX SEARCH

1. Idea

Mini-max algorithm is a recursive or backtracking algorithm which is used in decision-making and game theory. It provides an optimal move for the player assuming that opponent is also playing optimally. Mostly used for game playing in AI, such as Chess, Checkers, tic-tac-toe, go, and various tow-players game. This Algorithm computes the minimax decision for the current state. In this algorithm two players play the game, one is called MAX and other is called MIN. Both Players of the game are opponent of each other, where MAX will select the maximized value and MIN will select the minimized value. The minimax algorithm proceeds all the way down to the terminal node of the tree, then backtrack the tree as the recursion.

Pseudocode

```
Minimax(S, Maximizing=True)
      if S is a terminal state:
            return utility(S)
      if Maximizing:
            v = -infinity
            for a in actions(S):
                  v = max(v, Minimax(result(S, a), Maximizing=False))
            return v
      else:
            v = +infinity
            for a in actions(S):
                  v = min(v, Minimax(result(S, a), Maximizing=True))
            return v
```

2. Example

Demo Youtube video***:*** ***https://youtu.be/ZuCRG7fXcsI***

3. Complexity

- The time complexity of the Minimax algorithm is O (bm), where b represents the game tree's branching factor and m represents the maximum depth of the tree.
- The space complexity of minimax is O (bm).

4. Conclusion

✓ **Pros**

- Minimax is complete, as it definitely finds the solution (if existing) in the finite search tree.
- Always finds optimal solutions.
- Decision making in AI is easily possible.
- Easy to implement.

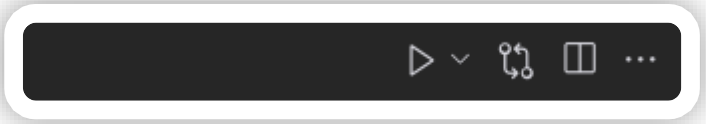
× **Cons**

- It has a huge branching factor, which makes the process of reaching the goal state slow.

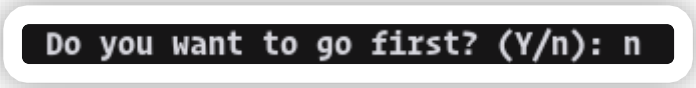
- Search and evaluation of unnecessary nodes or branches of the game tree degrades the overall performance and efficiency of the engine.
- Exploring the entire tree is not possible as there is a restriction of time and space.

III) TUTORIAL

1. **Run program**

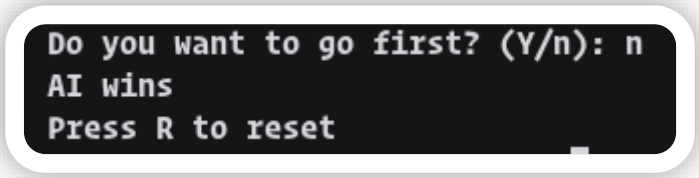


2. **Choose your turn**



3. **Program visualize the algorithms**

- If you wins(you can't btw), return "You wins".
- If AI wins, return "AI wins":



4. **Press R to replay**

IV) BRIEF DESCRIPTION

- Programming Language: Python
- Python package: pygame
- IDE: Visual Studio Code
- OS: Arch Linux

I'm also define color of X/O for visualization.

- Default: white
- Grid: light green
- X: green
- O: dark blue

Each function, variable are noted carefully as comment in my source code. This just is a brief, you need to read the source code for better understanding.

V) SUMMARY

Minimax Algorithm is one of the beneficial algorithms that have allowed researchers to traverse into new fields of smart and intelligent machines, that are capable of playing and winning games like tic-tac-toe, chess, checkers, Go, etc. against human opponents. Though not untouched by drawbacks, Minimax still remains an effective search algorithm whose advantages are further improved with the help of the Alpha Beta Algorithm. It may not be the best choice for the games with exceptionally high branching factor (e.g. game of GO). Nonetheless, given a proper implementation, it can be a pretty smart AI.

I'm sure there a some logical bugs(rarely AI choose not optimal solution and I'm seriously don't know where it go wrong) and I'm unable to fix it in less than 2 weeks, but I'm trying my best to finish this Adversarial Search project

Overall: 80/100

VI) REFERENCE

Visualize. Youtube. https://www.youtube.com/watch?v=Bk9hlNZc6sE&t=6084s
Minimax. Wiki. https://en.wikipedia.org/wiki/Minimax
Lectures sildes.